UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/726,902	12/03/2003	Mitchell Alsup	5500-88700	4177

53806 7590 03/31/2008

MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL (AMD)
P.O. BOX 398
AUSTIN, TX 78767-0398

EXAMINER	
FENNEMA, ROBERT E	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
03/31/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/726,902	ALSUP ET AL.	
	Examiner	Art Unit	
	ROBERT E. FENNEMA	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 January 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-35 have been considered.

2. In view of the Appeal Brief filed on 1/7/2008, PROSECUTION IS HEREBY REOPENED. New grounds of rejection are set forth below.

To avoid abandonment of the application, appellant must exercise one of the following two options:

(1) file a reply under 37 CFR 1.111 (if this Office action is non-final) or a reply under 37 CFR 1.113 (if this Office action is final); or,

(2) initiate a new appeal by filing a notice of appeal under 37 CFR 41.31 followed by an appeal brief under 37 CFR 41.37. The previously paid notice of appeal fee and appeal brief fee can be applied to the new appeal. If, however, the appeal fees set forth in 37 CFR 41.20 have been increased since they were previously paid, then appellant must pay the difference between the increased fees and the amount previously paid.

A Supervisory Patent Examiner (SPE) has approved of reopening prosecution by signing below:

.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the

invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-2, 11-15, 24-26, and 32-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg et al. (herein Rotenberg), in view of Xia.

5. As per Claim 1, Rotenberg teaches: A microprocessor (Abstract), comprising:

an instruction cache configured to store instructions (Section 2.1, first paragraph);

a branch prediction unit (Section 2.1, first paragraph);

a trace cache configured to store a plurality of traces of instructions (Section 2.2, second paragraph); and

a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache (Figure 4, and Section 2.2. The Instruction latch appears to fill the role of a prefetch unit);

wherein the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache); and

wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address (inherent in Rotenberg); and

wherein in response to the prefetch unit identifying a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of

the plurality of traces from the trace cache (Section 2.2, where it is said that "on a trace cache hit, an entire trace of instructions is fed into the instruction latch, bypassing the instruction cache), but fails to teach:

wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches. In addition, because traces would only thus be started on branches, it wouldn't make sense to look for a hit in the trace cache on non-branches, since it couldn't possibly result in a hit, thus one of ordinary skill in the art would not search a trace cache when it is not required, and save power by not having to constantly access the cache with no hope of a hit.

6. As per Claim 2, Rotenberg teaches: The microprocessor of claim 1, wherein the branch prediction unit is configured to output the predicted target address in response to

a prediction that a branch will be taken (Section 2.1, which discloses "They (the BTB banks) serve the role of detecting branches in the instructions currently being fetched and providing their target addresses, in time for the next fetch cycle. A few paragraphs further down, it is stated that this happens "if there is a predicted taken branch").

7. As per Claim 11, Rotenberg teaches: The microprocessor of claim 1, wherein each of the plurality of traces comprises partially-decoded instructions (it is inherent that a trace comprises a partially-decoded instruction, otherwise the necessary control information as show in section 2.2 would not be available).

8. As per Claim 12, Rotenberg teaches: The microprocessor of claim 1, wherein each of the plurality of traces is associated with a tag comprising the address of an earliest instruction, in program order, stored within that trace (Section 2.2, where the tag identifies the starting address of the trace).

9. As per Claim 13, Rotenberg teaches: The microprocessor of claim 1, wherein each of the plurality of traces is associated with a flow control field comprising a label for an instruction to which control will pass for each branch operation comprised in that trace (Section 2.2, the branch flags exist for every branch in the trace, and indicate which instruction control will pass to (the taken or not taken)).

10. As per Claim 14, Rotenberg teaches: A computer system, comprising:

a system memory (inherent if the system has an instruction cache); and

a microprocessor coupled to the system memory (also inherent in Rotenberg's invention), comprising:

an instruction cache configured to store instructions (Section 2.1, first paragraph);

a branch prediction unit (Section 2.1, first paragraph);

a trace cache configured to store a plurality of traces of instructions (Section 2.2); and

a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache (Figure 4, and Section 2.2. The Instruction latch appears to fill the role of a prefetch unit);

wherein the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache);

wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting the predicted target address (inherent in Rotenberg); and

wherein in response to the prefetch unit identifying a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the plurality of traces from the trace cache (Section 2.2, where it is said that "on a trace

cache hit, an entire trace of instructions is fed into the instruction latch, bypassing the instruction cache), but fails to teach:

wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs the predicted target address.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches. In addition, because traces would only thus be started on branches, it wouldn't make sense to look for a hit in the trace cache on non-branches, since it couldn't possibly result in a hit, thus one of ordinary skill in the art would not search a trace cache when it is not required, and save power by not having to constantly access the cache with no hope of a hit.

11. As per Claim 15, Rotenberg teaches: The computer system of claim 14, wherein the branch prediction unit is configured to output the predicted target address in response to a prediction that a branch will be taken (Section 2.1, which discloses "They

(the BTB banks) serve the role of detecting branches in the instructions currently being fetched and providing their target addresses, in time for the next fetch cycle. A few paragraphs further down, it is stated that this happens "if there is a predicted taken branch").

12. As per Claim 24, Rotenberg teaches: The computer system of claim 14, wherein each of the plurality of traces comprises partially-decoded instructions (it is inherent that a trace comprises a partially-decoded instruction, otherwise the necessary control information as show in section 2.2 would not be available).

13. As per Claim 25, Rotenberg teaches: The computer system of claim 14, wherein each of the plurality of traces is associated with a tag comprising the address of an earliest instruction, in program order, stored within that trace (Section 2.2, where the tag identifies the starting address of the trace).

14. As per Claim 26, Rotenberg teaches: The computer system of claim 14, wherein each of the plurality of traces is associated with a flow control field comprising a label for an instruction to which control will pass for each branch operation comprised in that trace (Section 2.2, the branch flags exist for every branch in the trace, and indicate which instruction control will pass to (the taken or not taken)).

15. As per Claim 32, Rotenberg teaches: A method, comprising:

fetching instructions from an instruction cache (Section 2.1, paragraphs 1-3);

continuing to fetch instructions from the instruction cache until a branch target address is generated (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache);

in response to a branch target address being generated, searching trace cache for an entry corresponding to the branch target address (Section 2.2, third paragraph), but fails to teach:

without searching a trace cache;

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not teach not searching the trace cache until a branch target address is generated, and in fact teaches that the trace cache is searched on every instruction. However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage of using less cache space, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches. An effect of only having traces start on branches is that it is then inherent that a non-branch instruction can never be the start of a trace, therefore there would be no reason to search the trace cache for a non-branch instruction, which has further potential advantages such as power saving and potential critical path reduction, as one of ordinary skill in the art

would recognize.

16. As per Claim 33, Rotenberg teaches: The method of claim 32, further comprising continuing to fetch instructions from the instruction cache in response to no entry being identified in the trace cache corresponding to the branch target address (Section 2.2, where it is said that "on a trace cache miss, fetching proceeds normally from the instruction cache).

17. As per Claim 34, Rotenberg teaches: The method of claim 32, further comprising fetching one or more traces from the trace cache in response to an entry being identified in the trace cache corresponding to the branch target address (Section 2.2, where it is said that "on a trace cache hit, an entire trace of instructions is fed into the instruction latch, bypassing the instruction cache).

18. Claims 3, 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg and Xia, in view of Patterson et al. (herein Patterson).

19. As per Claim 3, Rotenberg teaches the microprocessor of claim 1, but fails to teach: wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction. However, Patterson teaches that in order to reduce the penalty for a misprediction, you can fetch both the taken and not taken instructions, and put them in the BTB, which would then be able to

immediately output the correct path on a misprediction without having to fetch it (Pages 276-277). While it would increase the cost of the system, the advantage is a smaller misprediction penalty, which may worth the cost, depending on the needs of the system. Therefore, one of ordinary skill in the art at the time the invention was made would have stored both the taken and not taken branch paths in the BTB in order to reduce the misprediction penalty, and thus increasing performance (by allowing the other path to be immediately output when the misprediction is noted).

20. As per Claim 16, Rotenberg teaches the computer system of claim 14, but fails to teach: wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction. However, Patterson teaches that in order to reduce the penalty for a misprediction, you can fetch both the taken and not taken instructions, and put them in the BTB, which would then be able to immediately output the correct path on a misprediction without having to fetch it (Pages 276-277). While it would increase the cost of the system, the advantage is a smaller misprediction penalty, which may worth the cost, depending on the needs of the system. Therefore, one of ordinary skill in the art at the time the invention was made would have stored both the taken and not taken branch paths in the BTB in order to reduce the misprediction penalty, and thus increasing performance (by allowing the other path to be immediately output when the misprediction is noted).

21. Claims 4, 10, 17, 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg and Xia, in view of Braught, further in view of Xia.

22. As per Claim 4, Rotenberg teaches: The microprocessor of claim 1, further comprising a trace generator (it is necessary for Rotenberg's invention to have a trace generator in order to create traces), but fails to teach:

wherein the trace generator is configured to begin a trace with an instruction corresponding to a label boundary.

Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenberg's invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenberg's invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

23. As per Claim 10, Rotenberg teaches: The microprocessor of claim 4, wherein the trace generator is configured to generate traces in response to instructions being decoded (Section 2.2, the trace can not be completed (written into the cache) until the trace target addresses are calculated, which requires the instructions to be decoded).

24. As per Claim 17, Rotenberg teaches: The computer system of claim 14, further comprising a trace generator (it is necessary for Rotenberg's invention to have a trace generator in order to create traces), but fails to teach:

wherein the trace generator is configured to begin a trace with an instruction corresponding to a label boundary.

Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenberg's invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenberg's invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

25. As per Claim 23, Rotenberg teaches: The computer system of claim 17, wherein the trace generator is configured to generate traces in response to instructions being decoded (Section 2.2, the trace can not be completed (written into the cache) until the trace target addresses are calculated, which requires the instructions to be decoded).

26. Claims 5-8 and 18-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, Xia and Braught, further in view of Lange et al. (USPN 3,896,419, herein Lange).

27. As per Claim 5, Rotenberg, Xia and Braught teach the microprocessor of claim 4, but fail to teach:

wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces (because in order to trace multiple paths, every branch must be traced, even if it is a duplicate at the time, because it may not be a duplicate for future branches), and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation

(preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to

Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

28. As per Claim 6, Lange teaches: The microprocessor of claim 5, wherein in response to the trace generator identifying a duplicate copy of the trace, the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

29. As per Claim 7, Rotenberg teaches: The microprocessor of claim 5, wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated (Section 2.2. The trade cache is checked every time there is a potential new trace, so when one trace is found and discarded, the next potential new trace will cause the trace cache to be checked again).

30. As per Claim 8, Lange teaches: The microprocessor of claim 7, wherein in response to the trace generator identifying a trace entry corresponding to the next trace to be generated, the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

31. As per Claim 18, Rotenberg, Xia and Braught teach the computer system of claim 17, but fail to teach:

wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces (because in order to trace multiple paths, every branch must be traced, even if it is a duplicate at the time, because it may not be a duplicate for future branches), and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of

what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

32. As per Claim 19, Lange teaches: The computer system of claim 18, wherein in response to the trace generator identifying a duplicate copy of the trace, the trace

generator is configured to discard the trace under construction (Column 5, Lines 5-10).

33. As per Claim 20, Rotenberg teaches: The computer system of claim 18, wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated (Section 2.2. The trade cache is checked every time there is a potential new trace, so when one trace is found and discarded, the next potential new trace will cause the trace cache to be checked again).

34. As per Claim 21, Lange teaches: The computer system of claim 20, wherein in response to the trace generator identifying a trace entry corresponding to the next trace to be generated, the trace generator is configured to discard the trace under construction (Column 5, Lines 5-10).

35. Claims 9 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, Xia, and Braught, in view of Akkary et al. (USPN 6,247,121, herein Akkary).

36. As per Claim 9, Rotenberg teaches the microprocessor of claim 4, but fails to teach:

wherein the trace generator is configured to generate traces in response to instructions being retired.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3 that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

37. As per Claim 22, Rotenberg teaches the computer system of claim 17, but fails to teach:

wherein the trace generator is configured to generate traces in response to instructions being retired.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3

that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

38. Claims 27-31 and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg, Xia, Braught, and Lange, further in view of Akkary.

39. As per Claim 27, Rotenberg teaches: A method, comprising:

beginning construction of a new trace (Section 2.2), but fails to teach:

receiving a retired instruction; and

determining if a previous trace under construction duplicates a trace in a trace cache and if the received instruction corresponds to a branch label; and

beginning construction of the trace in response to determining that a previous trace under construction duplicates a trace in a trace cache and that the received instruction corresponds to a branch label.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite

requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches.

However, this combination still fails to teach starting a trace on a label boundary. Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenberg's invention, this would mean that a trace would be generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenberg's invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3

that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to

finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

The fact that traces can only begin on branch labels means that the next trace cannot be started until an instruction represents a branch label, regardless of when the duplication is found and dealt with.

40. As per claim 28, Rotenberg teaches: The method of claim 27, further comprising continuing construction of an incomplete trace already in process in response to determining that the incomplete trace does not duplicate a trace in the trace cache (Section 2.2).

41. As per Claim 29, Lange teaches: The method of claim 27, further comprising searching the trace cache for duplicate entries subsequent to completion of the previous trace under construction or the new trace (Column 5, Lines 5-10).

42. As per Claim 30, Rotenberg teaches: The method of claim 29, further comprising creating a new entry in the trace cache in response to no duplicate entry being identified (Section 2.2).

43. As per Claim 31, Lange teaches: The method of claim 29, further comprising discarding a trace in response to a duplicate entry being identified (Column 5, Lines 5-10).

44. As per Claim 35, Rotenberg teaches: A microprocessor comprising:

means for starting a new trace (Section 2.2), but fails to teach:

means for receiving a retired operation;

means for determining if a previous trace under construction duplicates a trace in a trace cache and if the received operation is a first operation at a branch label; and

means for starting a new trace in response to determining that a previous trace under construction duplicates a trace in trace cache and that the received operation is a first operation at a branch label.

Rotenberg teaches starting a trace on a trace cache miss, which then causes the line-fill buffer to begin filling (Section 2.2, fourth paragraph). Rotenberg does not specifically teach that an instruction must be a branch to access the cache (despite requiring branch prediction bits). However, Xia teaches that an advantage of starting a trace only at predictable branch instructions (Section 5.5, first scheme) is that less cache space is required, in direct contrast to Rotenberg, which teaches tracing on all instructions, requiring significantly more space (second scheme). Given this advantage, one of ordinary skill in the art at the time the invention was made would have been motivated by the advantage of needing less space for the trace cache to implement traces only on branches.

However, this combination still fails to teach starting a trace on a label boundary. Braught teaches that in Assembly Language, most branches operate on labels, which the machine can only interpret when converted to an address (Page 3). In Braughts example on page 3, all branches are labels, making every branch a label boundary. When combined with Rotenberg's invention, this would mean that a trace would be

generated with an instruction corresponding to a label boundary, as all the branches would go to labels, and even an address may be interpreted as a label. Given this knowledge, it would have been obvious to one of ordinary skill in the art that Rotenberg's invention begins traces when it encounters an instruction with a label boundary, as it only begins on branch instructions, which branch to labels.

While Rotenberg teaches that the instructions need to be decoded (Section 2.2) before the trace can be generated, it is not taught that they need to be retired beforehand (although it is suggested in Section 2.3, fill issues). Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (Column 3, Lines 40-44). Rotenberg discusses in Section 2.3 that some traces are committed but never used, thus evicting a useful trace. By ensuring that the trace is correct, the odds that it will be useful is increased, as an incorrect (not actually executed) trace should probably not need to be used, as branches tend to behave the same way (Section 1.1). Therefore, one of ordinary skill in the art at the time the invention was made would have waiting until an instruction was retired before considering adding it to the trace, in order to ensure accuracy of the trace and to prevent displacement of trace more likely to be reused.

Rotenberg teaches a standard trace cache, with the potential for some duplication, when considering the alternative embodiment of path associativity disclosed in Section 2.3. And while Rotenberg does not discuss the issue of duplication, he does teach of disadvantages which occur when a trace cache miss occurs while servicing a previous trace cache miss, and that this situation needs to be avoided, otherwise

features which increase cost or decrease performance must be implemented, and further teaches the disadvantages of a useless trace displacing a useful trace (Section 2.3, judicious trace selection). Therefore, Rotenberg teaches a system in which allows tracing of potential duplicate traces, and also a system which requires action when a miss occurs while servicing a miss. Lange teaches a system in which a cache is checked for a value while accessing memory, and if the identical value is found in the cache, the fetch to memory is aborted, freeing the memory to be used by a different operation (preventing a blocking load, Column 9, Lines 60-65) This is analogous to Rotenberg in the sense that Lange provides a motivation to look for duplicates in a cache to prevent a large delay and allow another operation to continue (analogous to finding a solution to what to do when encountering a miss while servicing a miss) by implementing a solution of searching for the duplicate, and not performing a long-latency operation on the duplicate value since it is not required (if the previous trace appears to be a duplicate of what is in the trace, there is no need to continue tracing, and the duplicate value can be discarded as taught by Lange (there is no need to perform an action on a duplicate value), preventing the line-fill buffer from blocking new traces, a long-latency event).

In an alternative viewpoint disclosed above, Rotenberg also teaches why it is disadvantageous to displace a useful trace with a useless trace (under the interpretation that a duplicate trace being stored in the trace cache is useless, as the original can always be accessed in its place), thus, with the potential for duplicate traces to exist with path associativity in Rotenberg's alternative embodiments, Rotenberg indicates in

his judicial trace selection that storing a duplicate trace would be at best useless, and at worst displace a useful trace that may be used, providing a motivation to prevent these duplicate traces from entering the trace cache from the line fill buffer, further providing a motivation to find some method to handle these cases, which Lange provides, by simply discarding the duplicate value.

Given these advantages in both of these situations, one of ordinary skill in the art at the time the invention was made would have been motivated to combine the teachings of Lange's checking for duplicates and discarding of the duplicates to Rotenberg, in order to solve either or both of the problems of getting a miss while servicing a miss, and preventing eviction of a useful trace by a duplicate trace.

The fact that traces can only begin on branch labels means that the next trace cannot be started until an instruction represents a branch label, regardless of when the duplication is found and dealt with.

Response to Arguments

45. Regarding Applicants arguments for Claim 32 (which has a rejection which has now been incorporated into the independent claims, which is why Examiner will address it now), Applicant has argued that Xia and Rotenberg both check the trace cache on every instruction, therefore there is a "direct conflict with the limitations of claim 32". However, Examiner is using Xia to teach the advantages of starting traces on branches, and is not attempting to bodily incorporate the two references, as Applicant appears to be arguing, which is improper. Xia teaches a reason as to why one would only want to

start a trace on a branch (smaller cache size), and Examiner has said that one of ordinary skill in the art would be able to take that knowledge, and when incorporating it into Rotenberg, realize that if traces only start on branches, that it is impossible to get a trace cache hit when the instruction is not a branch, thus it would be a waste of power to even try (and may also cause critical path problems, issues that those of ordinary skill in the art would clearly be able to recognize, and thus would implement the invention as claimed, because it is an obvious modification based on the information provided by the references. Examiner reminds Applicant that the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981), and Examiner asserts that the combined teachings of the references would have suggested to one of ordinary skill in the art the limitations of the claim.

46. Regarding Applicants arguments for Claim 11 and 24, Applicant has argued that the Examiners assertion that "traces inherently comprise partially-decoded instructions" is clearly incorrect. However, these traces clearly hold data that cannot be obtained without some form of decoding, for example, the traces hold the start and next addresses of the trace, information which is impossible to obtain without decoding, since an instruction is just a random collection of bits until it is decoded and put into a

context. The branch flags and branch mask bits also require decoding, since simply identifying that there is even one branch in the trace requires that that branch instruction was decoded to identify it as a branch.

47. Regarding Applicants arguments for Claims 13 and 26, Examiner has clarified the rejection, which can be seen in the rejection themselves.

48. Regarding Applicants arguments for Claim 3, Applicant has argued that the limitation of "wherein the branch prediction unit is configured to output the predicted target address in response to detection of a branch misprediction" is not taught in the references used for the rejection. However, Examiner notes that in the combination proposed, the branch predictor holds both the taken and not taken paths, therefore, on a misprediction, the other path is available to be output immediately, thus teaching the limitation.

49. Regarding Applicants arguments for Claim 10, Examiner refers to his remarks for Claim 11, which shows that the trace requires decoded instructions in order to be created, thus this limitation is taught.

50. Regarding Applicants arguments for Claims 5 and 18, Applicant has argued that the references do not teach "wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that is being constructed", and has argued

that Rotenberg does not allow for duplicate traces, thus cannot check for them. However, as Examiner has argued previously, Rotenberg does have the potential to have duplicate traces, in Section 2.3, when tracing multiple paths, the trace under construction could be a duplicate of a currently existing trace, and there is no way to know until it is done, since with path associativity, the trace could be 99% identical to the previous trace, and differ only on the final branch, thus it would be a duplicate until it arrived at that branch, and if it turned out to not be different, the trace under construction would be a duplicate (the system MUST trace everything in this embodiment, even with a cache hit, because it will not know if it is a duplicate or not until the end). As Examiner has brought up this embodiment previously, Examiner asks Applicant to provide an argument as to why this embodiment/interpretation is unreasonable, as opposed to arguing that Rotenberg cannot have duplicate traces, when Examiner has provided evidence that he does.

51. Regarding Applicants arguments towards Claims 6-7 and 19-20, Examiner believes the remarks set forth above also deal with these claims sufficiently.

52. Regarding Applicants arguments towards Claim 8, Lange teaches an analogous situation to the claim, where a duplicate copy of a value in a cache is discarded when it is found to be a duplicate, which is what the claim is claiming. Additionally, Examiner has already laid out a motivation for combining Lange with Rotenberg, and Examiner refers the Applicant to the rejection to see said motivation.

53. Regarding Applicants arguments for Claim 9, Applicant is attempting to bodily incorporate the references, as opposed to looking at the rejection that the Examiner has laid out. Akkary teaches that retired instructions are guaranteed to be correct, and Rotenberg teaches that putting incorrect instructions into the trace cache can cause significant problems, therefore the Examiner has said it would have been obvious to one of ordinary skill in the art to wait until an instruction is retired in order to place it in the trace cache, and the manner in which Akkary uses a trace buffer is completely unrelated to the rejection, and it is improper to bodily incorporate the references as the Applicant is attempting to do to argue the rejection of the claim.

54. Applicants remaining arguments are rehashes of arguments already presented, and Examiner refers to the above responses in order to address those similar claims.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to ROBERT E. FENNEMA whose telephone number is (571)272-2748. The examiner can normally be reached on Monday-Friday, 8:30-6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Eddie P Chan/
Supervisory Patent Examiner, Art Unit 2183

Robert E Fennema
Examiner
Art Unit 2183

RF